Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'tput.1' command

$ man tput.1

tput(1) General Commands Manual tput(1)

NAME

tput, reset - initialize a terminal or query terminfo database

SYNOPSIS

tput [-Ttype] capname [parameters]

tput [-Ttype] [-x] clear

tput [-Ttype] init

tput [-Ttype] reset

tput [-Ttype] longname

tput -S <<

tput -V

DESCRIPTION

The tput utility uses the terminfo database to make the values of ter?

minal-dependent capabilities and information available to the shell

(see sh(1)), to initialize or reset the terminal, or return the long

name of the requested terminal type. The result depends upon the capa?

bility's type:

string

tput writes the string to the standard output. No trailing

newline is supplied.

integer

tput writes the decimal value to the standard output, with a

trailing newline.

boolean

tput simply sets the exit code (0 for TRUE if the terminal has the capability, 1 for FALSE if it does not), and writes nothing to the standard output.

Before using a value returned on the standard output, the application should test the exit code (e.g., $?, see sh(1)) to be sure it is 0. (See the EXIT CODES and DIAGNOSTICS sections.) For a complete list of capabilities and the capname associated with each, see terminfo(5).

Options

-S allows more than one capability per invocation of tput. The ca? pabilities must be passed to tput from the standard input in? stead of from the command line (see example). Only one capname is allowed per line. The -S option changes the meaning of the 0 and 1 boolean and string exit codes (see the EXIT CODES sec? tion).

Because some capabilities may use string parameters rather than numbers, tput uses a table and the presence of parameters in its input to decide whether to use tparm(3X), and how to interpret the parameters.

-Ttype indicates the type of terminal. Normally this option is unnec? essary, because the default is taken from the environment vari? able TERM. If -T is specified, then the shell variables LINES and COLUMNS will also be ignored.

-V reports the version of ncurses which was used in this program, and exits.

-x do not attempt to clear the terminal's scrollback buffer using the extended ?E3? capability.

Commands

A few commands (init, reset and longname) are special; they are defined by the tput program. The others are the names of capabilities from the terminal database (see terminfo(5) for a list). Although init and re? set resemble capability names, tput uses several capabilities to per? form these special functions.

capname

indicates the capability from the terminal database.

If the capability is a string that takes parameters, the argu? ments following the capability will be used as parameters for the string.

Most parameters are numbers. Only a few terminal capabilities require string parameters; tput uses a table to decide which to pass as strings. Normally tput uses tparm(3X) to perform the substitution. If no parameters are given for the capability, tput writes the string without performing the substitution.

init If the terminal database is present and an entry for the user's terminal exists (see -Ttype, above), the following will occur:

(1) first, tput retrieves the current terminal mode settings for your terminal. It does this by successively testing

? the standard error,

? standard output,

? standard input and

? ultimately ?/dev/tty?

to obtain terminal settings. Having retrieved these set? tings, tput remembers which file descriptor to use when up? dating settings.

(2) if the window size cannot be obtained from the operating system, but the terminal description (or environment, e.g., LINES and COLUMNS variables specify this), update the oper? ating system's notion of the window size.

(3) the terminal modes will be updated:

? any delays (e.g., newline) specified in the entry will be set in the tty driver,

? tabs expansion will be turned on or off according to the specification in the entry, and

? if tabs are not expanded, standard tabs will be set (every 8 spaces).

(4) if present, the terminal's initialization strings will be

output as detailed in the terminfo(5) section on Tabs and Initialization,

(5) output is flushed.

If an entry does not contain the information needed for any of these activities, that activity will silently be skipped.

reset This is similar to init, with two differences:

(1) before any other initialization, the terminal modes will be reset to a ?sane? state:

? set cooked and echo modes,

? turn off cbreak and raw modes,

? turn on newline translation and

? reset any unset special characters to their default values

(2) Instead of putting out initialization strings, the termi? nal's reset strings will be output if present (rs1, rs2, rs3, rf). If the reset strings are not present, but ini? tialization strings are, the initialization strings will be output.

Otherwise, reset acts identically to init.

longname

If the terminal database is present and an entry for the user's terminal exists (see -Ttype above), then the long name of the terminal will be put out. The long name is the last name in the first line of the terminal's description in the terminfo data? base [see term(5)].

Aliases

tput handles the clear, init and reset commands specially: it allows for the possibility that it is invoked by a link with those names.

If tput is invoked by a link named reset, this has the same effect as tput reset. The tset(1) utility also treats a link named reset spe? cially.

Before ncurses 6.1, the two utilities were different from each other:

? tset utility reset the terminal modes and special characters (not

done with tput).

? On the other hand, tset's repertoire of terminal capabilities for resetting the terminal was more limited, i.e., only reset_1string, reset_2string and reset_file in contrast to the tab-stops and mar? gins which are set by this utility.

? The reset program is usually an alias for tset, because of this difference with resetting terminal modes and special characters.

With the changes made for ncurses 6.1, the reset feature of the two programs is (mostly) the same. A few differences remain:

? The tset program waits one second when resetting, in case it hap? pens to be a hardware terminal.

? The two programs write the terminal initialization strings to dif? ferent streams (i.e., the standard error for tset and the standard output for tput).

Note: although these programs write to different streams, redirect? ing their output to a file will capture only part of their actions.

The changes to the terminal modes are not affected by redirecting the output.

If tput is invoked by a link named init, this has the same effect as tput init. Again, you are less likely to use that link because another program named init has a more well-established use.

Terminal Size

Besides the special commands (e.g., clear), tput treats certain ter? minfo capabilities specially: lines and cols. tput calls setupterm(3X) to obtain the terminal size:

? first, it gets the size from the terminal database (which generally is not provided for terminal emulators which do not have a fixed window size)

? then it asks the operating system for the terminal's size (which generally works, unless connecting via a serial line which does not support NAWS: negotiations about window size).

? finally, it inspects the environment variables LINES and COLUMNS which may override the terminal size.

If the -T option is given tput ignores the environment variables by calling use_tioctl(TRUE), relying upon the operating system (or fi? nally, the terminal database).

EXAMPLES

tput init

Initialize the terminal according to the type of terminal in the environmental variable TERM. This command should be included in everyone's .profile after the environmental variable TERM has been exported, as illustrated on the profile(5) manual page.

tput -T5620 reset

Reset an AT&T 5620 terminal, overriding the type of terminal in the environmental variable TERM.

tput cup 0 0

Send the sequence to move the cursor to row 0, column 0 (the upper left corner of the screen, usually known as the ?home? cursor po? sition).

tput clear

Echo the clear-screen sequence for the current terminal.

tput cols

Print the number of columns for the current terminal.

tput -T450 cols

Print the number of columns for the 450 terminal.

bold=`tput smso` offbold=`tput rmso`

Set the shell variables bold, to begin stand-out mode sequence, and offbold, to end standout mode sequence, for the current termi? nal. This might be followed by a prompt: echo "${bold}Please type in your name: ${offbold}\c"

tput hc

Set exit code to indicate if the current terminal is a hard copy terminal.

tput cup 23 4

Send the sequence to move the cursor to row 23, column 4.

tput cup

Send the terminfo string for cursor-movement, with no parameters substituted.

tput longname

Print the long name from the terminfo database for the type of terminal specified in the environmental variable TERM.

tput -S <<!

> clear

> cup 10 10

> bold

> !

This example shows tput processing several capabilities in one in? vocation. It clears the screen, moves the cursor to position 10, 10 and turns on bold (extra bright) mode. The list is terminated by an exclamation mark (!) on a line by itself.

FILES

/usr/share/terminfo

compiled terminal description database

/usr/share/tabset/*

tab settings for some terminals, in a format appropriate to be output to the terminal (escape sequences that set margins and tabs); for more information, see the Tabs and Initialization, section of terminfo(5)

EXIT CODES

If the -S option is used, tput checks for errors from each line, and if any errors are found, will set the exit code to 4 plus the number of lines with errors. If no errors are found, the exit code is 0. No in? dication of which line failed can be given so exit code 1 will never appear. Exit codes 2, 3, and 4 retain their usual interpretation. If the -S option is not used, the exit code depends on the type of cap? name:

boolean

a value of 0 is set for TRUE and 1 for FALSE.

string a value of 0 is set if the capname is defined for this termi?

nal type (the value of capname is returned on standard out?

put); a value of 1 is set if capname is not defined for this

terminal type (nothing is written to standard output).

integer

a value of 0 is always set, whether or not capname is defined

for this terminal type. To determine if capname is defined

for this terminal type, the user must test the value written

to standard output. A value of -1 means that capname is not

defined for this terminal type.

other reset or init may fail to find their respective files. In

that case, the exit code is set to 4 + errno.

Any other exit code indicates an error; see the DIAGNOSTICS section.

DIAGNOSTICS

tput prints the following error messages and sets the corresponding

exit codes.

exit code error message

??

0 (capname is a numeric variable that is not specified in

the terminfo(5) database for this terminal type, e.g.

tput -T450 lines and tput -T2621 xmc)

1 no error message is printed, see the EXIT CODES section.

2 usage error

3 unknown terminal type or no terminfo database

4 unknown terminfo capability capname

>4 error occurred in -S

??

HISTORY

The tput command was begun by Bill Joy in 1980. The initial version

only cleared the screen.

AT&T System V provided a different tput command, whose init and reset

subcommands (more than half the program) were incorporated from the re?

set feature of BSD tset written by Eric Allman.

Keith Bostic replaced the BSD tput command in 1989 with a new implemen?

tation based on the AT&T System V program tput. Like the AT&T program, Bostic's version accepted some parameters named for terminfo capabili? ties (clear, init, longname and reset). However (because he had only termcap available), it accepted termcap names for other capabilities. Also, Bostic's BSD tput did not modify the terminal I/O modes as the earlier BSD tset had done.

At the same time, Bostic added a shell script named ?clear?, which used tput to clear the screen.

Both of these appeared in 4.4BSD, becoming the ?modern? BSD implementa? tion of tput.

This implementation of tput began from a different source than AT&T or BSD: Ross Ridge's mytinfo package, published on comp.sources.unix in December 1992. Ridge's program made more sophisticated use of the ter? minal capabilities than the BSD program. Eric Raymond used that tput program (and other parts of mytinfo) in ncurses in June 1995. Using the portions dealing with terminal capabilities almost without change, Raymond made improvements to the way the command-line parameters were handled.

PORTABILITY

This implementation of tput differs from AT&T tput in two important ar? eas:

? tput capname writes to the standard output. That need not be a regular terminal. However, the subcommands which manipulate termi? nal modes may not use the standard output.

The AT&T implementation's init and reset commands use the BSD (4.1c) tset source, which manipulates terminal modes. It succes? sively tries standard output, standard error, standard input before falling back to ?/dev/tty? and finally just assumes a 1200Bd termi? nal. When updating terminal modes, it ignores errors.

Until changes made after ncurses 6.0, tput did not modify terminal modes. tput now uses a similar scheme, using functions shared with tset (and ultimately based on the 4.4BSD tset). If it is not able to open a terminal, e.g., when running in cron, tput will return an

error.

? AT&T tput guesses the type of its capname operands by seeing if all of the characters are numeric, or not.

Most implementations which provide support for capname operands use the tparm function to expand parameters in it. That function ex? pects a mixture of numeric and string parameters, requiring tput to know which type to use.

This implementation uses a table to determine the parameter types for the standard capname operands, and an internal library function to analyze nonstandard capname operands.

This implementation (unlike others) can accept both termcap and ter? minfo names for the capname feature, if termcap support is compiled in. However, the predefined termcap and terminfo names have two ambiguities in this case (and the terminfo name is assumed):

? The termcap name dl corresponds to the terminfo name dl1 (delete one line).

The terminfo name dl corresponds to the termcap name DL (delete a given number of lines).

? The termcap name ed corresponds to the terminfo name rmdc (end delete mode).

The terminfo name ed corresponds to the termcap name cd (clear to end of screen).

The longname and -S options, and the parameter-substitution features used in the cup example, were not supported in BSD curses before 4.3reno (1989) or in AT&T/USL curses before SVr4 (1988).

IEEE Std 1003.1/The Open Group Base Specifications Issue 7 (POSIX.1-2008) documents only the operands for clear, init and reset. There are a few interesting observations to make regarding that:

? In this implementation, clear is part of the capname support. The others (init and longname) do not correspond to terminal capabili? ties.

? Other implementations of tput on SVr4-based systems such as So? laris, IRIX64 and HPUX as well as others such as AIX and Tru64 pro?

vide support for capname operands.

? A few platforms such as FreeBSD recognize termcap names rather than terminfo capability names in their respective tput commands. Since 2010, NetBSD's tput uses terminfo names. Before that, it (like FreeBSD) recognized termcap names.

Beginning in 2021, FreeBSD uses the ncurses tput, configured for both terminfo (tested first) and termcap (as a fallback).

Because (apparently) all of the certified Unix systems support the full set of capability names, the reasoning for documenting only a few may not be apparent.

? X/Open Curses Issue 7 documents tput differently, with capname and the other features used in this implementation.

? That is, there are two standards for tput: POSIX (a subset) and X/Open Curses (the full implementation). POSIX documents a subset to avoid the complication of including X/Open Curses and the termi? nal capabilities database.

? While it is certainly possible to write a tput program without us? ing curses, none of the systems which have a curses implementation provide a tput utility which does not provide the capname feature.

X/Open Curses Issue 7 (2009) is the first version to document utili? ties. However that part of X/Open Curses does not follow existing practice (i.e., Unix features documented in SVID 3):

? It assigns exit code 4 to ?invalid operand?, which may be the same as unknown capability. For instance, the source code for Solaris' xcurses uses the term ?invalid? in this case.

? It assigns exit code 255 to a numeric variable that is not speci? fied in the terminfo database. That likely is a documentation er? ror, confusing the -1 written to the standard output for an absent or cancelled numeric value versus an (unsigned) exit code.

The various Unix systems (AIX, HPUX, Solaris) use the same exit-codes as ncurses.

NetBSD curses documents different exit codes which do not correspond to either ncurses or X/Open.

SEE ALSO

clear(1), stty(1), tabs(1), tset(1), curs_termcap(3X), terminfo(5).

This describes ncurses version 6.2 (patch 20210508).